Mining negative generalized knowledge from relational databases	Kỹ thuật khai phá tri thức tổng quát hóa phủ định từ cơ sở dữ liệu quan hệ

checked Attribute-oriented induction (AOI) is a useful data mining method for extracting generalized knowledge from relational data and users' background knowledge. Concept hierarchies can be integrated with the AOI method to induce multi-level generalized knowledge. However, the existing AOI approaches are only capable of mining positive knowledge from databases; thus, rare but important negative generalized knowledge that is unknown, unexpected, or contradictory to what the user believes, can be missed. In this study, we propose a global negative attribute-oriented induction (GNAOI) approach that can generate comprehensive and multiple-level negative generalized knowledge at the same time. Two pruning properties, the downward level closure property and the upward superset closure property, are employed to improve the efficiency of the algorithm, and a new interest measure, nim(cl), is exploited to measure the degree of the negative relation. Experiment results from a real-life dataset show that the proposed method is effective in finding global negative generalized knowledge.	Quy nạp hướng thuộc tính (AIO) là một phương pháp khai phá dữ liệu trích xuất tri thức tổng quát từ dữ liệu quan hệ và cơ sở tri thức nền tảng của người dùng. Phân cấp khái niệm có thể tích hợp với phương pháp AIO để tạo ra tri thức tổng quát đa mức. Tuy nhiên, những phương pháp AIO hiện tại chỉ có khả năng khai thác tri thức khẳng định từ các cơ sở dữ liệu; vì thế những tri thức tổng quát phủ định hiếm nhưng quan trọng chẳng hạn như những tri thức chưa được biết đến, bất ngờ, hoặc trái ngược với niềm tin của người dùng có thể bị bỏ qua. Trong nghiên cứu này, chúng tôi đề xuất phương pháp quy nạp hướng thuộc tính phủ định toàn cục (GNAIO) có thể tạo ra tri thức tổng quát phủ định đa mức và toàn diện cùng một lúc. Hai tính chất cắt tỉa, tính chất đóng mức dưới và tính chất đóng superset trên được dùng để cải thiện hiệu suất của thuật toán, và một đại lượng mới đặc trưng cho mức độ quan tâm, nim(cl), được khai thác để đo mức độ của tương quan nghịch. Kết quả thực nghiệm từ tập dữ liệu thực tế chứng tỏ rằng phương pháp do chúng tôi đề xuất hiệu quả trong việc tìm kiếm tri thức tổng quát phủ định toàn cục.
1. Introduction The information explosion has been a serious challenge for cur-rent information gathering institutions. There is an urgent need for improved technology that can access, analyze,	

summarize, and interpret information intelligently and automatically [1]. Data mining has been proposed in response to this demand, and many varied approaches have been developed [2-5]. Data generalization, which generalizes a large set of task-relevant data from a relatively low conceptual level to higher conceptual levels, is one of the most important approaches for summarizing data characterization. Such a technique can generate concise and abstract knowledge, which helps in providing overall pictures of the data at different levels [6,7]. We call such data generalized knowledge.

Attribute-oriented induction (AOI), which was first introduced by Cai et al. [8], is a data mining technique used to extract generalized knowledge from relational data and users' background knowledge. This essential background knowledge takes the form of a concept hierarchy that is associated with each attribute in a relational database [9]. A concept hierarchy often refers to domain knowledge, and is a way to store relationships between specific concepts and generalized concepts [10]. The integration of the concept hierarchies and AOI methods can allow us to induce multi-level and cross-level generalized knowledge, which can provide good decision making support for different levels of management.

There is no doubt that the AOI technique is very useful for find-ing

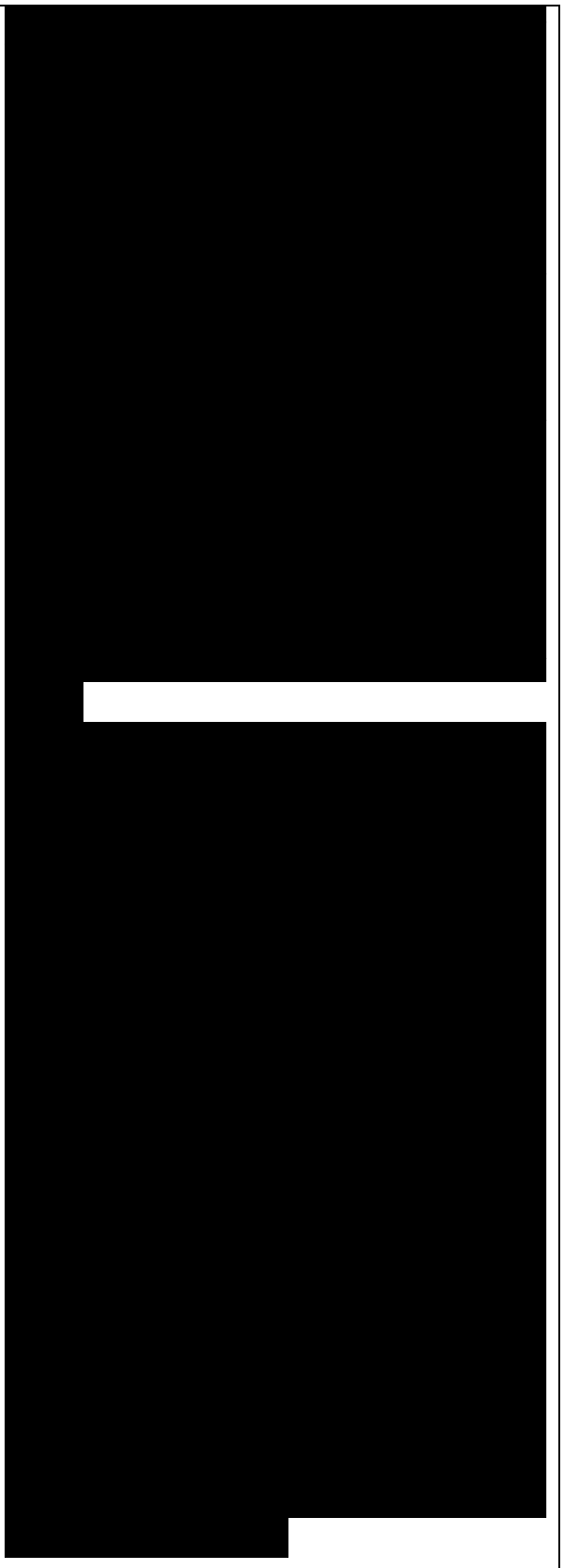
general characteristics from a relational database, and it has been applied in many areas, such as spatial patterns [11,12], med-ical science [13,14], intrusion detection [15,16], strategy making [17], and financial prediction [18]. However, as we can see from these applications, existing AOI approaches are only focused on mining positive generalized knowledge from databases. In the real world, there is a type of knowledge that is unknown, unexpected, or contradictory to what a user believes, called negative general-ized knowledge in this study, which is more novel, meaningful, and interesting to the user than positive facts.

Thus, discovering negative generalized knowledge from a rela-tional database can reveal more interesting phenomena. For exam-ple, a medical database may tell us that although many patients are Taiwanese and numerous patients have H1N1, only a few Tai-wanese have this disease. This interesting negative fact suggests that, due to some unknown reasons, Taiwanese are more resistant to H1N1. However, this negative fact can never be found using traditional approaches, which only find positive relations or associations between data.

In this work, a generalized tuple is defined as an association of attribute values in concept hierarchies. For

example, (Taiwanese, Pneumonia) and (Taiwanese, H1N1) are examples of generalized tuples. A generalized tuple is negative if its actual occurrence is significantly lower than its expected occurrence. The aim of this paper is to find all negative generalized tuples. To the best of our knowledge, the mining of negative tuples has been neglected in previous research on AOI; instead, the previous studies mined only positive generalized facts. As a result of the low frequency of negative generalized tuples, which might contradict our expectations, these tuples may never be found.

To resolve this problem, this study proposed a novel approach, called global negative AOI (GNAOI), which can generate comprehensive and multiple-level negative generalized knowledge at the same time. Only one pass over a database is necessary with the developed algorithm, and the efficiency of the algorithm is improved by the employment of two pruning properties, the downward level closure property and the upward superset closure property, and a new interest measure, nim(cl), is exploited to measure the negative degree. Furthermore, it is not appropriate to generate all negative generalized tuples without information reduction, because too much information may hinder users from applying knowledge. Therefore, we must come up with some way of finding only the interesting negative



generalized tuples. Accordingly, we use the following criteria to search for interesting negative generalized tuples:

(1) Minimum support and expected support constraint: Each value in a negative generalized tuple must satisfy the minimum support constraint. Meanwhile, a negative generalized tuple must satisfy the minimum expected support constraint. The rationale is that if a generalized tuple covers only a tiny frac-tion of data, it is more likely to represent noise rather than some other more interest characteristic. Therefore, we require that each generalized tuple must satisfy a minimum representability constraint.

(2) Strong negative association: A negative tuple represents a large deviation between the actual and the expected sup-ports of a tuple. Therefore, we say two values have strong negative association if the probability of their co-occurrence is significantly smaller than the multiplication of their indi-vidual probabilities. The deviation of a negative generalized tuple is measured by the relative ratio of its actual support and its expected support occurrence. Only those generalized tuples with strong deviations are worth keeping.

(3) Redundant measure constraint: The multiple-level taxonomic structures means that a generalized tuple can induce many similar tuples simply by replacing some attribute values with their ancestors or descendants in the taxonomy.

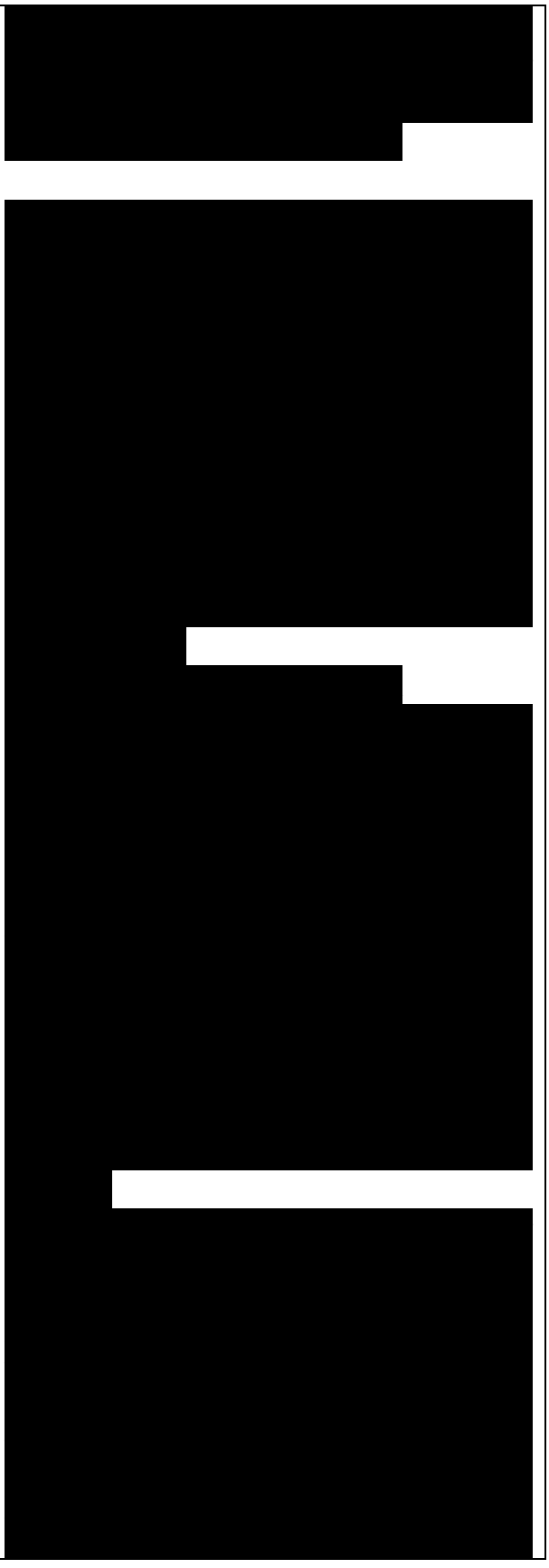
Accordingly, many generalized tuples become redundant because their information can be reasonably induced from other tuples; such tuples are unnecessary and should be removed.

The remainder of the paper is organized as follows. In Section 2, the AOI method and related work are reviewed. In Section 3, we define the problem of mining global negative generalized knowledge from a relational database. In Section 4, the mining algorithm, GNAOI, is presented. The results of our performance evaluation are given in Section 5. Finally, some conclusions are drawn in Section 6.

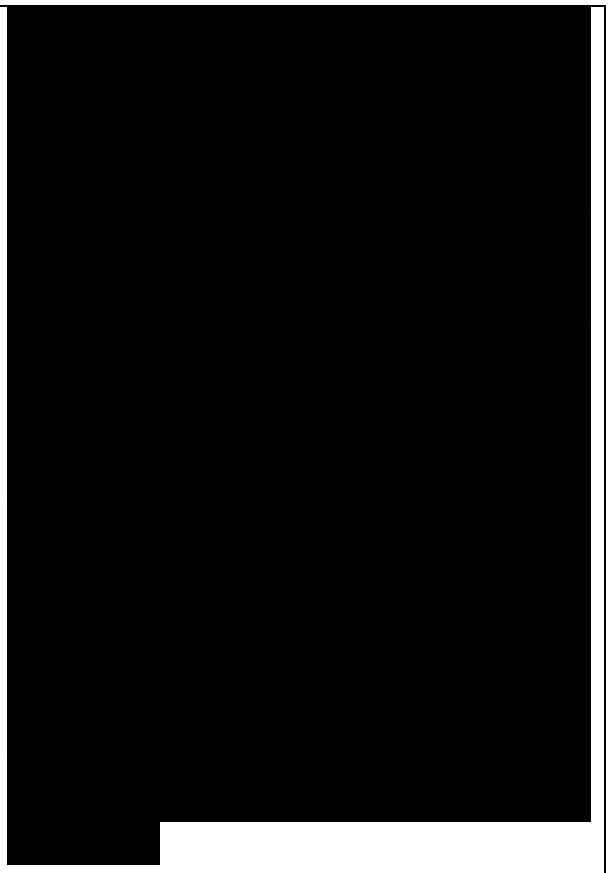
2. Related work

AOI is a set-oriented database mining method that generalizes the task-relevant subsets of the data, attribute by attribute [19]. The generalization of each attribute is associated with a concept hierarchy. The concept hierarchies represent necessary back-ground knowledge, which controls the generalization process and can range from the single, most generalized root concept, to the most specific concepts corresponding to the specific values of attri-butes in the database [10,20].

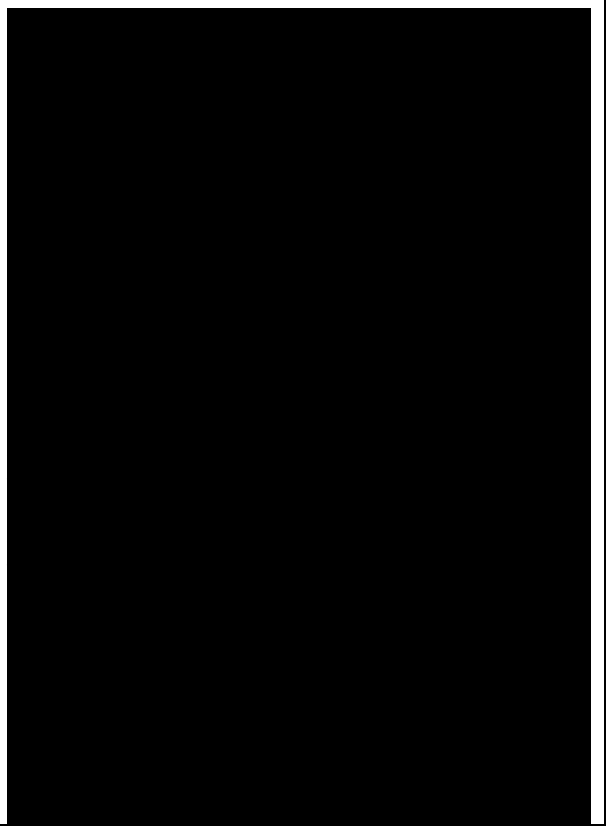
There has been much research effort made using AOI ap-proaches. Carter and Hamilton [21] proposed more efficient meth-ods for AOI, while Cheung [22] used a rule-based conditional concept hierarchy to extend the traditional approach to a condi-tional AOI, thereby allowing different tuples to be generalized through



different paths, depending on other attributes of the tu-ples. Hsu [23] extended the basic AOI algorithm to the generaliza-tion of numeric values, and Chen and Shen [20] proposed a dynamic programming algorithm based on AOI techniques to find generalized knowledge from an ordered list of data. Several inde-pendent groups of researchers have combined fuzzy concept hier-archy procedures with AOI [2,24-26]. A fuzzy hierarchy of concepts reflects the degree to which a concept belongs to its direct parent concept. More than one direct parent of a single concept is allowed during fuzzy induction. AOI has also been integrated with other applications to induce domain dependent generalized knowledge [11-18,27,28].



The above methods, however, all concentrate on mining positive generalized knowledge. To the best of our knowledge, the mining of multi-level negative generalized tuples has not been addressed. The only studies similar to our work are those concerned with negative association rule mining methods. According to the classification scheme proposed by Tan et al. [29] there are three approaches for the mining of negative association rules or patterns. The first notes the absence of a certain item as a negative item [30,31]. After selecting a small set of negative items that we are interested in, we can augment the database by adding these negative items to corresponding transactions. In this way,



existing association rule mining methods can be used to find negative association rules. However, as pointed out in Tan et al. [29], the major weakness of this approach lies in the fact that it can only handle a small number of negative items. When the number of negative items is large, the computation time becomes terribly long.

The second type is a neighborhood-based approach known as indirect association [32,33]. A pair of items a and b, is said to have indirect association if there is an item set Y such that support(a, Y) and support(b, Y) are both high, while support(a, b) is low. In other words, a and b are replacements for each other in the sense that they are seldom purchased together, but at the same time, there are a number of other items commonly purchased together with them. Although this approach is interesting, it is incapable of finding negative associations between items at multiple-levels in hierarchies.

The third type relies on a concept hierarchy to estimate the ex-pected support of an itemset. In this way, the methods can deter-mine whether an itemset is negative or not [34,35]. Although the methods also use concept hierarchy to estimate the expected sup-ports of the itemsets, there are some important differences be-tween this method and our approach:

(1) Their databases are transactional, while ours is relational.

(2) They use a single hierarchy for the entire database, but we set a hierarchy for every attribute.

(3) The outputs are different. They generate negative associa-tion rules, but we generate negative generalized tuples.

(4) Their negative association rules are composed of original items, while our negative generalized tuples are composed of original items as well as generalized items from the hierarchies.

(5) They use a concept hierarchy to estimate the expected sup-ports of itemsets rather than generating patterns with gen-eralized items in the hierarchies. In this work, we do both.

In short, we take a fundamentally different approach to that which has been discussed thus far. Our goal is to develop an effi-cient mining method for discovering negative knowledge across different levels of taxonomies from relational databases.

3. Problem formulation

In this section, we define the problem of mining global negative generalized tuples from a relational database. Let A = {a1, a2 an} be a set of attributes. The value domain of each attribute ai is

associated with a corresponding concept hierarchy tree Ti. A concept hierarchy represents a taxonomy of values for an attribute domain that are partially ordered according to a specific-to-abstract relation. A value in a tree is also called a node. Let vk,i denote the kth node of tree Ti, where the sequence of nodes is numbered according to some tree traversal order. Let lvl(vk,i) denote the level of value vk i; let leaf(Ti) denote the set of leaf nodes of tree Ti; and let nonleaf(Ti) denote the set of non-leaf nodes of tree Ti.

Example 1. Let us consider a small research grant database with 20 tuples and 4 attributes, as shown in Table 1. To save space, all concept hierarchies are shown only in Appendix A. In Fig. A1 in Appendix A, there are four attributes with four concept hierarchies, T1, T2, T3 and T4, respectively. Suppose the node labels are numbered according to the level order traversal. Now, we can find the values v10,2 = Spanish, v3,2 = 'Liberal arts' in hierarchy T2 as shown in Fig. 1. Moreover, we have lvl(v12) = 1, lvl(v9,2) = 3, leaf(T2) = {History, Music, Spanish, French Software_eng, Information_tech}.

Based on these notations, the problem can be stated as follows:

Definition 1 (Valueset). A valueset l = {vk1i1, vk2i2 vkrir} is a non-empty set of values, where no two values in the set belong to the same

attribute, i.e., all ip — iq. A valueset with k values is called a k-valueset.

For example, l = {Delhi, History} is a 2-valueset, while {Delhi, New York} is not.

Definition 2 (Tuple, Rid). Let m be the number of attributes in the relation. A primitive tuple t is an m-valueset whose each value is in leaf(Ti) for some i. Further, a generalized tuple g is an m-valueset whose each value is in leaf(Ti) or nonleaf(Ti) for some i. A relational database D is a set of primitive tuples. Each tuple in database D is associated with a unique identifier Rid.

Definition 3 (Cover). For each value vki, let cov(vk,,) denote the cover of value vk,i, which is the set of Rids of primitive tuples in D whose values of attribute i are vk,i or descendents of vk,i in the concept hierarchy Ti. For each valueset l = {vk1i1i vk2 i2vkrir}, cov(l)
= cov(vk1,i1) n ... n cov(vkr,lr) (1 6 r 6 n).
Example 2. In Table 1, each record is called a primitive tuple and has a unique number Rid. From Table 1, we find that cov(New York) = {2, 8, 12, 20}, cov(Physics) = {8, 11, 12} and cov{New York, Physics} = {8, 12}. After performing the generalized process, the cover of the abstract value is the union of the cover of all its descendents in the concept hierarchy

tree. Therefore, we have cov(USA) = {2, 5, 8, 11, 12, 17, 19, 20}, cov(Science) = {2, 8, 11, 12, 14} and cov(USA, Science) = {2, 8, 11, 12}.

Definition 4 (Support). The support of a valueset l in a database D, sup(l), is the number of Rids in cov(l) versus the total number of tuples in D, as shown in Eq. (1):

|cov(l)| ...

Sup(l)=L-j ■ (1) Example 3. The cardinality of |cov(USA, Science)| = |{2, 8, 11, 12}| = 4 and |D| = 20. So, we obtain sup(USA, Science) = 4/20 = 20%.

Real data tends to be dirty, noisy and error-prone and a lot of noise may be hidden in the data. Therefore if the frequency of some value among the values in the database is too small it is more likely to be a noise rather than information. If the support of a value is too small such values as noise and pruned to reduce the search space. Hence, a minValueSup threshold is necessary to ensure that the generalized tuple is statistically significant.

Definition 5 (minValueSup). A value vki is called a frequent value (fv) if sup(vk,i) p minValueSup as specified by the user. A candidate valueset (cl) is a non-empty valueset, where each value in the set is a frequent value.

Example 4. Consider the relation shown in Table 1. Assume that minValueSup = 15%. Table 2 shows partial results for the values at all levels, where the right hand column

indicates whether these values are in set FV or frequent values. We keep the values with support p minValueSup (15% in this example). The cover field records the Rids of those primitive tuples in Table 1 whose values are descendents or the same as the table values in the concept hierarchy. Furthermore, all the values are sorted from abstract concepts to specific concepts.

In addition to the minValueSup constraint, the other pruning threshold is minExpectedSup. An expected support is the expected probability that a valueset would occur if all values in the valueset are independent. In probability theory, two values X and Y are independent if P(X u Y) = P(X) x P(Y). According to the theory, we can define the expected support of cl as follows:

exSup(cl) = sup(vk1,h)x sup(vk2,i2)x---x sup(vkr,ir)(1 6 r 6 n)■

Since we are searching for negative generalized tuples whose actual frequencies are much lower than expected ones, we must set a minExpectedSup threshold to filter out the valuesets which cover only a tiny fraction of the data. By setting the threshold of minExpectedSup properly, we can remove many unimportant valuesets, and speed up the computation.

Definition 6 (minExpectedSup). A potential valueset (pl) is a cl whose expected support is not less than

minExpectedSup threshold.

Table 1

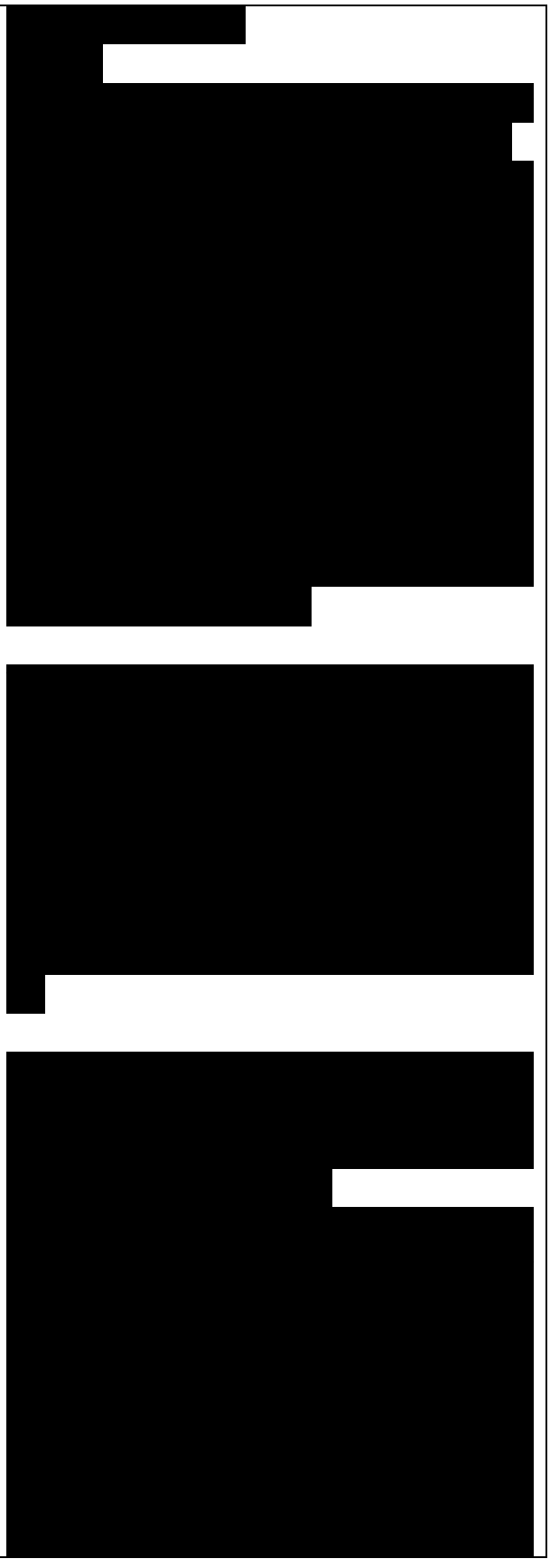
Synthetic relation showing research grants with 20 tuples and 4 attributes.

In Section 1, we mentioned that an interesting negative generalized tuple must satisfy three additional constraints. Obvi-ously, the potential valuesets in Definitions 5 and 6 satisfy the first constraint, including the minimal value support threshold and the minimal expected support threshold. In order to satisfy the second constraint, the strong negative association constraint, we must define the interest measure of a negative generalized tuple.

As mentioned above, a negative tuple represents a large deviation between the actual and the expected support of the tuple. The larger the deviation, the more interesting the tuple is considered to be [36]. Hence, the interest measure of a valueset is defined in terms of the unexpectedness of the valueset.

Definition 7 (Interest, minlnterest). The negative interest measure of pl, nim(pl), is obtained by exSup(pl) sup (pi) '

This measure is greater than one when the actual support value is lower than the expected support value, indicating a negative dependence. The larger the value is above 1, the greater is the negative dependence. A potential valueset pl is interesting if nim(pl) p minlnterest, where minlnterest is a user-specified constant. An interesting



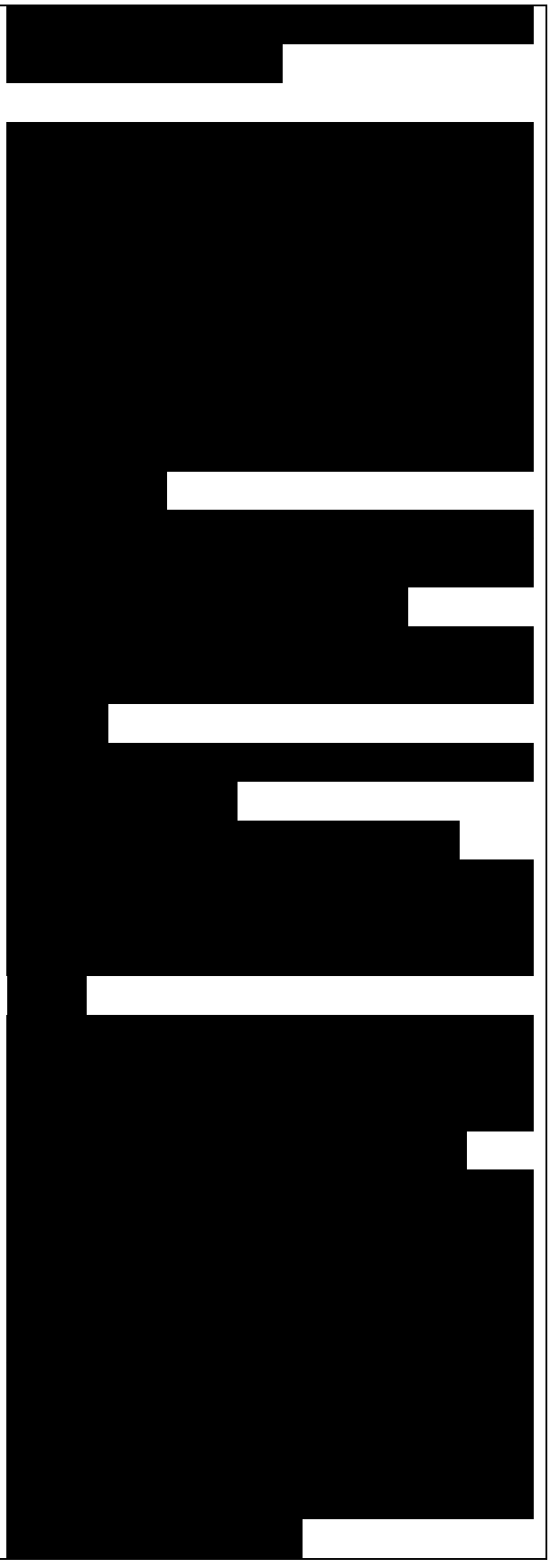
negative generalized tuple ng is an interesting negative m-valueset.

Example 5. Following Example 4, Table 3 shows the partial results for the candidate valueset (cl). Assume that minExpectedSup = 2% and minlnterest = 2.5. The last valueset cl4 is not a potential value- set, because exSup(cl4) is less than minExpectedSup. Moreover, only the first valueset is interesting, because nim(cl1) p minlnterest.

In addition to the pruning and interest constraints, two redundant cases need to be considered.

Definition 8 (Redundant valueset). Case 1: A valueset pla = {vk1i1, vk2,i2vkr,ir, vkr+1ir+1} is redundant if there is a valueset plb = {Vk1,i1, vk2,i2 Vkr,ir}, where nim(plb) p nim(pla). Case 2: A valueset pla = {vk1i1, vk2 i2 vkrir} is redundant if there is a value-set pl'a = {vk1>n, vk2>l2,..., vkrr}, where every value in pla is a descendant or the same as the corresponding value in pla, and nim(pl'a) P nim(pla).

Example 6. Suppose there are four negative valuesets, as shown in Table 4. The valueset pl1 is a subset of valueset pl2, and nim(pl2) < - nim(pl1). We regard pl2 as redundant and it should be removed. Another case is for the last two valuesets. Valueset pl3 is more abstract than pl4, and nim(pl3) p nim(pl4). We keep the more gen-eral valueset in this case.



3.1. Problem statement

Given a relational database D and a set of thresholds (minValue- Sup, minExpectedSup, minInterest), the aim is to find all non-redun- dant interesting negative generalized tuples.

4. Global negative attribute-oriented induction

In this section, we propose a novel GNAOI approach for mining multiple-level negative generalized tuples from a relational data-base. The algorithm combines attribute values with their concept hierarchies to induce the negative generalized knowledge. A hierar- chy-information encoded table [37], which stores the generalized identifier (GID) code of the corresponding attribute value and cover lists [38] (i.e., Rid-lists) is adopted to save space and speed up the computation. Furthermore, both the downward level closure property and upward superset closure property are employed in the algorithm to efficiently generate global negative generalized knowledge. The input of the GNAOI algorithm includes three parts: (1) a task-related relation; (2) a set of concept hierarchies; and (3) a set of thresholds. The output consists of non-redundant interesting negative generalized tuples learned from the relational data. The major steps of the GNAOI are shown in Fig. 2. We first discuss how to collect and transform the input relational data into the hierarchical-information encoded table. Then we discuss in detail the algorithm designed for inducing all negative

generalized tuples. Finally, the pruning and transformation processes are shown.

4.1. Collect and encode the task-relevant tuples

GNAOI first collects the dataset that is relevant to the learning task using relational database operations, e.g., projection and selec-

Table 3
Partial results for candidate valueset (CL).

Table 4

tion. The retrieved task-relevant tuples are stored in a table called the initial relation table. Each tuple in the initial relation has a unique identifier called Rid. Next, the initial relation can be trans-formed into a hierarchical-information encoded table, which stores the GID code of the corresponding attribute value. A GID is a number string containing the route information for an attribute value in a concept hierarchy.

Example 7. In Table 1, the amount attribute is the fourth attribute in the tuple. Suppose the value of the amount is ‘19800’. This can be encoded as ‘4121’, where the first digit ‘4’ represents the fourth attribute in tuple, the second ‘1’ represents the ‘Low’ at level-1, the third ‘2’ represents ‘15000-24999’ at level-2, and the fourth ‘1’ indicates ‘15000-19999’ at level-3 (Fig. 3). checked

It is beneficial to transform the original object values to GID codes. First, an encoded string contains complete route

informa-tion for an attribute value which provides a self-explanatory advantage. Second, a code requires fewer bits than the corre-sponding object-identifier which reduces the size of the tuples. Encoding can also be performed during the collection of task-relevant data, so there is no extra encoding pass required. The taxonomic information for each value in Table 1 is encoded as a GID code, following the encoding rule mentioned above. The results are shown in Table 5 (T).

4.2. The NGT algorithm

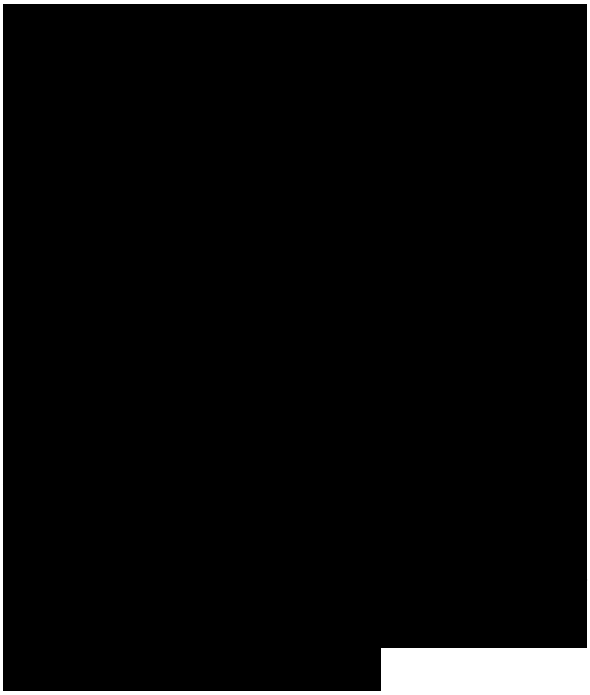
The generalization of negative valuesets is a difficult task and the computational load is heavy. How to effectively identify poten-tial negative tuples is the key problem. In this study, several useful constraints are used to generate and measure the potential value- sets. Both the downward level closure property and upward super-set closure property are included in the algorithm to improve the generalization efficiency. The first step in this algorithm is to scan the encoded table T, and find the set of all values (CV) for all levels. Next, the set of frequent values (FV) for all levels is generated, as defined in Definition 5, and sorted by a specific rule.

Let VSk be the set of negative k-valuesets. All k-valuesets (k p 2) are generated using VSk_1 and FV. The k-valueset generation is repeated until k is equal to the number of attributes in the tuple. It is important to note that 1-

valuesets (VS1) is the same as FV. Finally, we obtain all negative generalized tuples by uniting all k-valuesets. The details of the NGT algorithm are shown in Fig. 4.

First, a pass is made over T to produce the set of all values for all levels CV. Moreover, the support for each value at all levels is calculated. The cover (rids-list) is recorded at the same time by scanning T once. Using the cover technique, it is not necessary during the k-valuesets generalization process to scan the dataset any more. All frequent values of all levels (set FV) are generated in line 2. The steps can be formulated in Fig. 5.

The values in FV are then sorted according to their attribute and concept level, from abstract to specific. For example, if there is a FV ={11**, 1111, 12**, 111*, 112*, 122*, 1222}, then the sorted FV = {11**, 111*, 1111, 112*, 12**, 122*, 1222}. The sorting is beneficial for the subsequent generalization. It is important to note that the sorting process can also be performed at the init-pass (line 1) step by adopting appropriate data structure; no extra sorting pass is required. Furthermore, the FV is the same as 1-valuesets (VS1) which are used to generate the 2-valuesets.



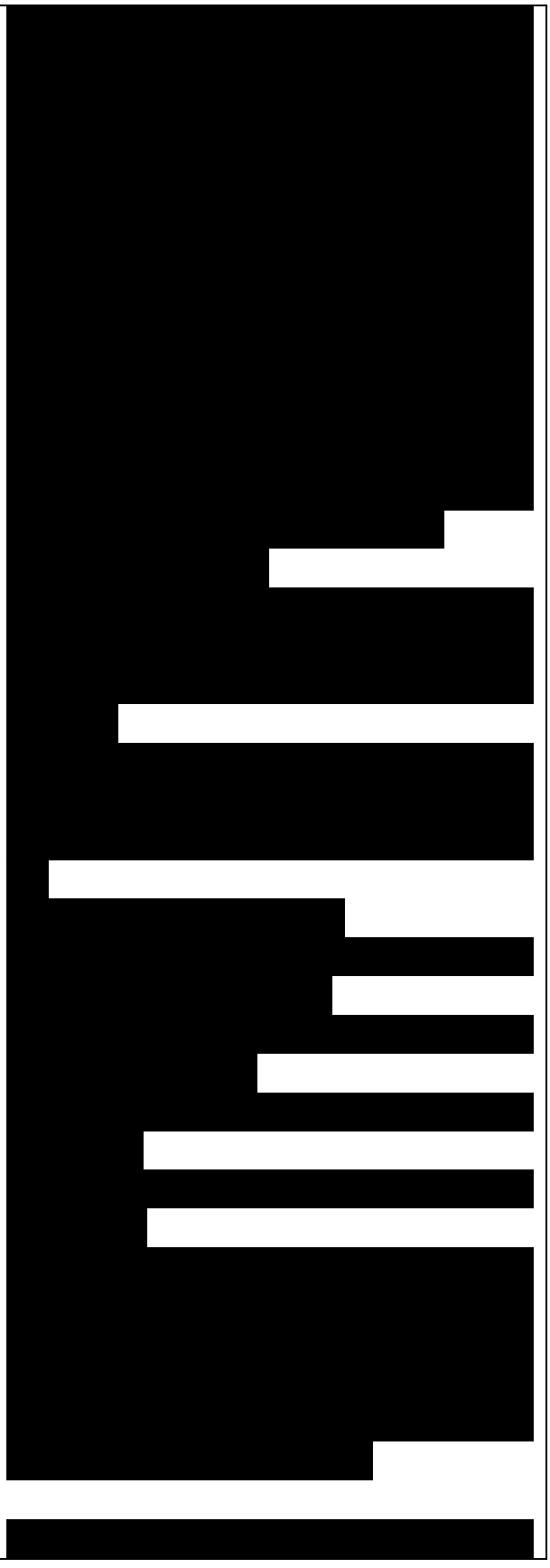
After finding the 1-valuesets, lines 5-7 proceed with the deriva-tion of the negative k-valuesets (k p 2). The

negative valuesets in set VSk_1 (found in the (k—1)th pass) and the frequent values in set FV are used to generate the negative valuesets VSk in each subsequent pass, say pass k, using the function negative-gen. In this function, the deviation between the actual and the expected support of each valueset is calculated. After this, two useful properties called downward level closure property and upward superset closure property are adopted to prune unnecessary candidates.

Algorithm GNAOI.
Input: (1) A task-related relation; (2) a set of concept hierarchies; (3) a set of thresholds.

Output. Non-redundant interesting negative generalized tuples mined from the task-related relation.

Method. Four steps:
Step 1. Collect and encode task-relevant tuples.
Step 2. Discover all negative generalized tuples.
Step 3. Remove redundant negative generalized tuples.
Step 4. Transform and Output final tuples.
Lemma 1 (Downward level closure property). If a k-valueset {v1, v2 vk} is not a potential valueset, then all of the valuesets f v1, v'2,..., vk}, where v\ is either vi or a descendant of vi, are not potential valuesets.
Proof. The support of a value is always



larger on the abstract level than that on the specific level That is, the higher the concept level, the larger the expected support. Consequently, if a valueset is not a potential valueset at the higher level, then its valuesets at the lower level are also not.

Therefore, the lemma follows.
For example, if {11**, 32**, 221*} is not a potential valueset, then {111*, 321*, 221*}, {112*, 32**, 2212}, {11**, 3213, 221*}, {1123, 3211, 2212} are also not potential valuesets. □

Lemma 2 (Upward superset closure property). If a k-valueset {v1, v2 vk} is not a potential valueset, then all of its (k + 1)-supersets are not potential valuesets.

Proof. The expected support of a k-valueset is always larger than that of its (k + 1)-superset. That is, the more values in the valueset, the smaller the expected support. Consequently, if a valueset is not a potential valueset, then none of its supersets are either. Therefore, the lemma follows.

For example, if {11 **, 32**} is not a potential valueset, then {11**, 32**, 221*}, {11**, 32**, 21**, 42**} are also not potential valuesets.

Fig. 6 gives the negative-gen algorithm, which takes VSk_1 and FV as arguments, and returns VSk, the set of all negative k-valuesets. The join

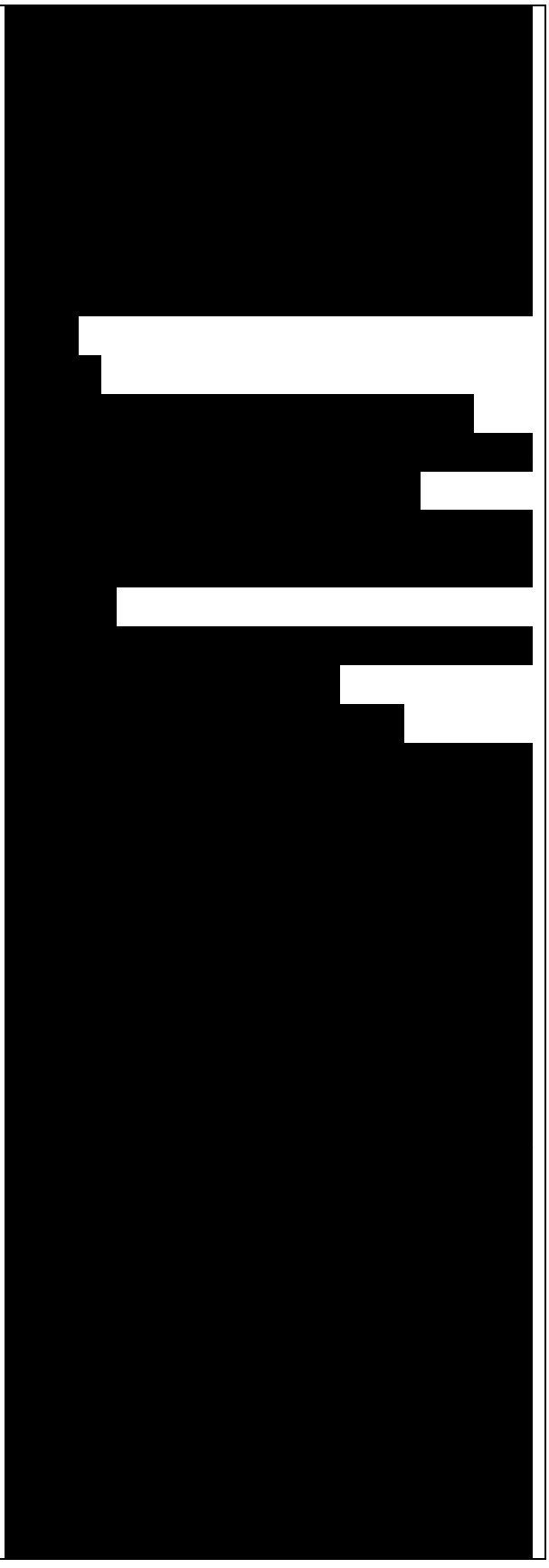
step uses a double loop (line 1-3) to generate the k-valuesets. However, each valueset in the VSk_1 contains different attribute values. Before generating a new valueset, the algorithm must find the other attributes which do not exist in the valueset. It is a nontrivial task to find these attributes and join them with existent

Table 5
Results encoded from Table 1: T

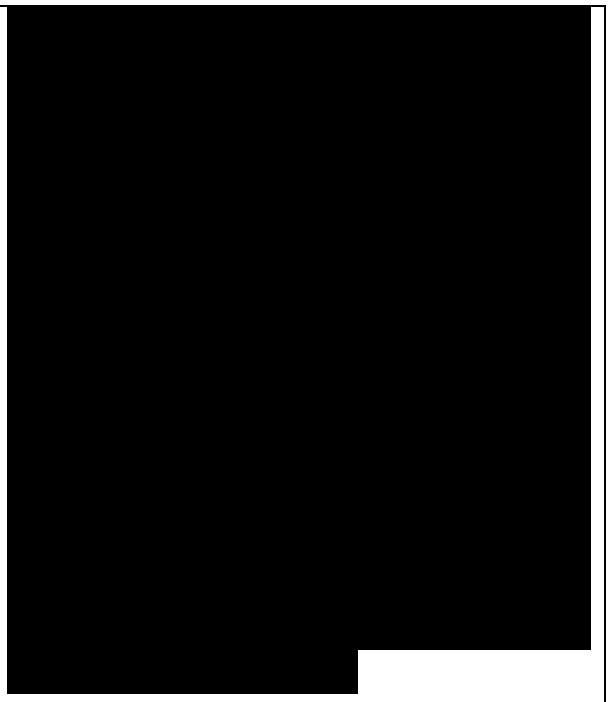
Algorithm (NGT) For discovering all negative generalized tuples.
Input. (1) The hierarchical-information encoded table T; and (2) the set of thresholds.
Output. All interesting negative generalized tuples.

Fig. 6. Negative-gen algorithm.

valuesets. For efficient joining, the FV is split into several FV(n), in which the argument n represents the attribute. For example, FV(1) = {11**, 111*, 1111, 112*, 12**, 122*, 1222...}, and FV(2) = {22**, 221*, 2212, 222*...}. The data required for the FV can be structured as a dynamic array list or a tree structure. After the join step, the algorithm examines whether the new valueset is marked, which will be ignored directly. On the other hand, line 5 calculates the expected support of valueset (u, v). As mentioned in Definition 6, if exSup((u, v)) is less than the minExpectedSup threshold, valueset (u, v) will be discarded (line 6) and no more the superset of valueset (u, v) will be generated. Further, according to the downward level closure property, all of the descendent k-valuesets of



valueset (u, v) will be also discarded (line 14). On the other hand, if exSup((u, v)) is larger than minExpectedSup, lines 7-9 are used to calculate the interest measure. Finally, the algorithm keeps the k-valuesets whose nim((u, v)) measures are larger than the minlnterest threshold (lines 10 and 11). Here, we also check if (u, v) matches the condition of redundant case 1 in Definition 8, by examining if there exists any subset of (u, v) with a stronger negative interest value. If so, (u, v) is redundant. Otherwise, it is insert into VSk.



After performing the negative-gen function, the NGT algorithm unites all k-valuesets, where k is from 2 to |attr| (line 8, Fig. 4). We thus obtain all negative generalized tuples. The final step is to convert all negative k-valuesets to the generalized tuple (line 9, Fig. 4). The step is given in Fig. 7. □

Example 8. The 2-valuesets {1231, 21**} can be converted to the generalized tuple {1231, 21**, any, any}, while the 3-valuesets {211*, 31**, 42**} can be converted to the generalized tuple {211*, 31**, 42**, any}.

Fig. 8. Redundant-pruning algorithm.

Table 6

Negative generalized tuples generated by the GNAOI algorithm.

4.3. Pruning and transformation

The negative generalized tuples discovered by the NGT algo-rithm include multiple-level values.

However, not every generalized tuple is worth being presented to the users. If a generalized tuple is more interesting than a tuple which is more specific (every value in the tuple is the same or at a lower level), the latter is considered a redundant tuple, as defined in Definition 8, Case 2. Using the algorithm shown in Fig. 8, all redundant generalized tuples can be removed. Since the interest value measures the negative depen-dence, the algorithm first sorts the generalized tuples by their interest value, from larger to smaller. The removal process is per-formed starting from the tuple which has the largest interest.

The final step is to transform and output all non-redundant negative generalized tuples. To do so, each GID code ofg is converted to an original value. Next, the final tuples are output. For example, the encoded tuple (111*, 211*, Any, 41**) will be converted to (India, Liberal arts, Any, Low).

5. Experiments

In this section, we discuss several experiments conducted to evaluate the performance of the proposed algorithm. The GNAOI algorithm was implemented using the Java programming language and tested on a PC with an Intel Core Duo 3.0 GHz processor and 2 GB main memory under the Windows XP operating system. A real dataset of credit card data was used in the experiments. To make the time measurements more reliable, no other application was

running on the machine while the experiments were running.

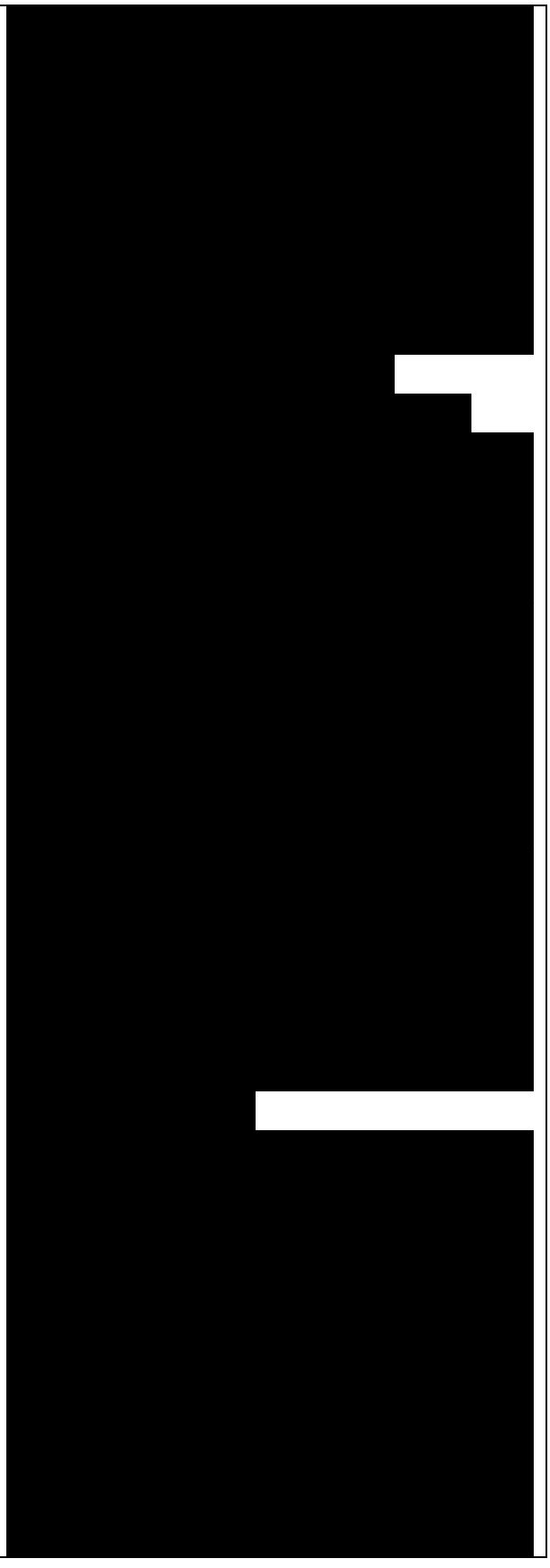
Since the traditional AOI method does not focus on negative tuples, the objective of this experiment was not to compare the two algorithms. Instead, we focused on evaluating the effectiveness and performance of the GNAOI algorithm under the various parameter settings. Specifically, its performance was studied with respect to four factors, including: (1) the number of tuples; (2) the number of attributes; (3) the minimal expected support thresholds; and (4) the interest thresholds. The run time and the variation of the number of negative generalized tuples with the change in value of the factors were noted.

5.1. Real dataset

The real dataset used in this study consists of credit card data collected from the survey research center at a university in Taiwan.

The dataset consists of 2 million tuples, each of which has 27 attri-butes including card type, frequency, sex, age, occupation, marital status, education, blood type, income, constellation, consumption pattern, and so on. We invited two domain experts who select six attributes whose values are well suited to repeated generalization and create concept hierarchies from them. The selected attributes are education, age, occupation, constellation, income and credit card consumption. The concept hierarchy for each attribute is shown in

Fig. A2 in Appendix A. Since the attribute values of ori-ginal dataset are stored as specific codes defined by the research center, we must first transform these attribute values to the leaf concepts of concept hierarchies, i.e., the GID code, before the gen-eralization tasks can be started.



5.2. Negative generalized tuples

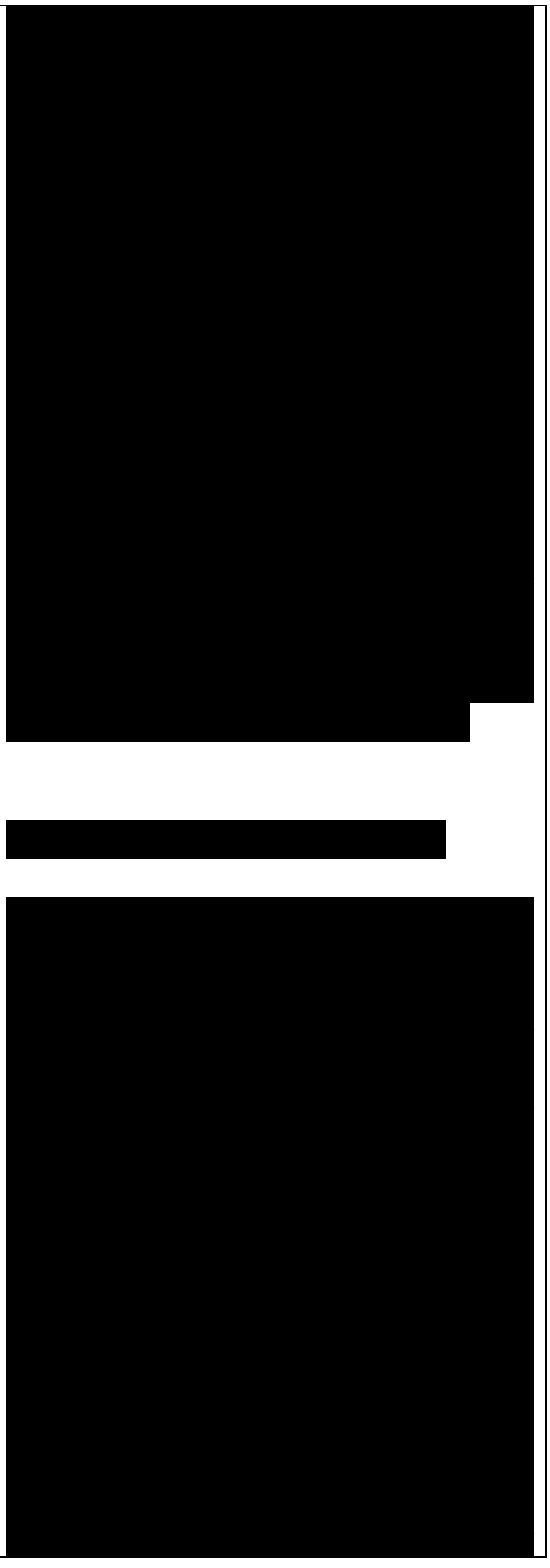
In this section, we discuss the negative generalized knowledge produced by the GNAOI algorithm. In the test, we randomly ex-tracted 50000 tuples, each with 6 attributes, from the credit card dataset to mine the negative generalized tuples. The parameters are as follows: minValueSup = 10%, minExpectedSup = 2% and the interest measure threshold is 1.2. Applying the GNAOI algorithm and repeating the extraction and mining process 50 times, we get the average amount of interesting negative generalized tuples is 637. Some interesting negative tuples are shown in Table 6. The values ‘Any’ are represented by ‘-’.

Each of the tuples in Table 6 can be interpreted as negative generalized knowledge. The table shows some interesting observa-tions. From tuples 1, 2, and 3, we see that the card consumption of full-time housewives with low incomes may be higher than ex-pected. One possible reason could be that since housewives usually manage a whole household they spend

more money with credit cards. Tuple 5 shows young-adult sales may earn low income but spending more money. Tuples 7 and 8 show advanced-age home workers are seldom high education, especially between 35 and 49 years old. Tuples 11 and 12 indicate that older students are conservative about spending money with credit cards. Tuples 13 and 14 show that older students seldom have high income, especially in the 35-49 age bracket. From tuples 15, 16 and 17, we see that young-adult workers with the Air element constellation and low income may spend more money with credit cards. In contrast, tuples 18 and 19 show that the Earth element constellation group is conservative at spending money with credit cards. Tuple 20 points out those younger high-level workers with high income are seldom low education.

5.3. Performance evaluations

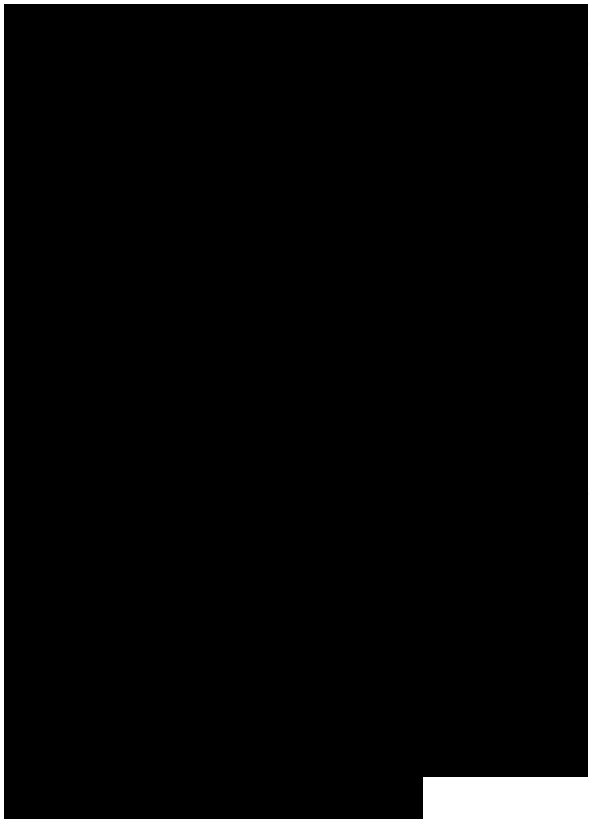
In this section, we describe a series of experiments evaluating the performance of the process. In the first, we evaluate the run time for the GNAOI algorithm using varied data size (number of tuples) from 5,000 to 50,000 but fixed all thresholds. The thresh-olds are as follows: minValueSup = 10%, minExpectedSup = 2%, and the interest measure threshold is 1.2. In addition, different numbers of attributes are evaluated: 4 attributes (attr-4); 5 attri-butes (attr-5); and 6 attributes (attr-6). We repeat the evaluations 50 times. Each time,



the evaluated datasets are randomly extracted from source database. The average results of computation are summarized in Fig. 9. From this figure, we can see that the time required increases as the number of tuples increases. Moreover, when more attributes are given, more run time is required. This is because the GNAOI algorithm generates the candidate valuesets by recursively combining distinct attribute values. Hence, the number of attributes has a significant effect on the run time.

We also study how the number of generalized tuples changes as the data size changes. Fig. 10 shows the results of filtering for dif-ferent data sizes and different numbers of attributes. The results show that more negative generalized tuples are generated as more attributes are considered. The reason is that more attributes result in more negative generalized tuples, which in turn generates more interesting tuples. However, the number of generalized tuples re-mains almost the same for the same number of attributes, even though the data size changes. The reason for this may be that, since the data are all drawn from the same dataset, the data sets have similar properties, even though their sizes are different.

Next, we study how the minExpectedSup threshold influences the run time of the GNAOI algorithm. To this end, we set five thresholds and fix the data size at 25000 and the number of attri-butes at 6. The other



thresholds are set as follows: minValueSup = 10% and the interest measure threshold is 1.2. We also repeat this evaluation 50 times. Each time, we randomly extracted 25000 tuples from source database. Fig. 11 shows the average per-formance curve of computation. Time required decreases as the ex-pected support threshold values increase. The reason is that when the thresholds increase, the number of candidate valuesets de-crease, which in turn decreases the run time.

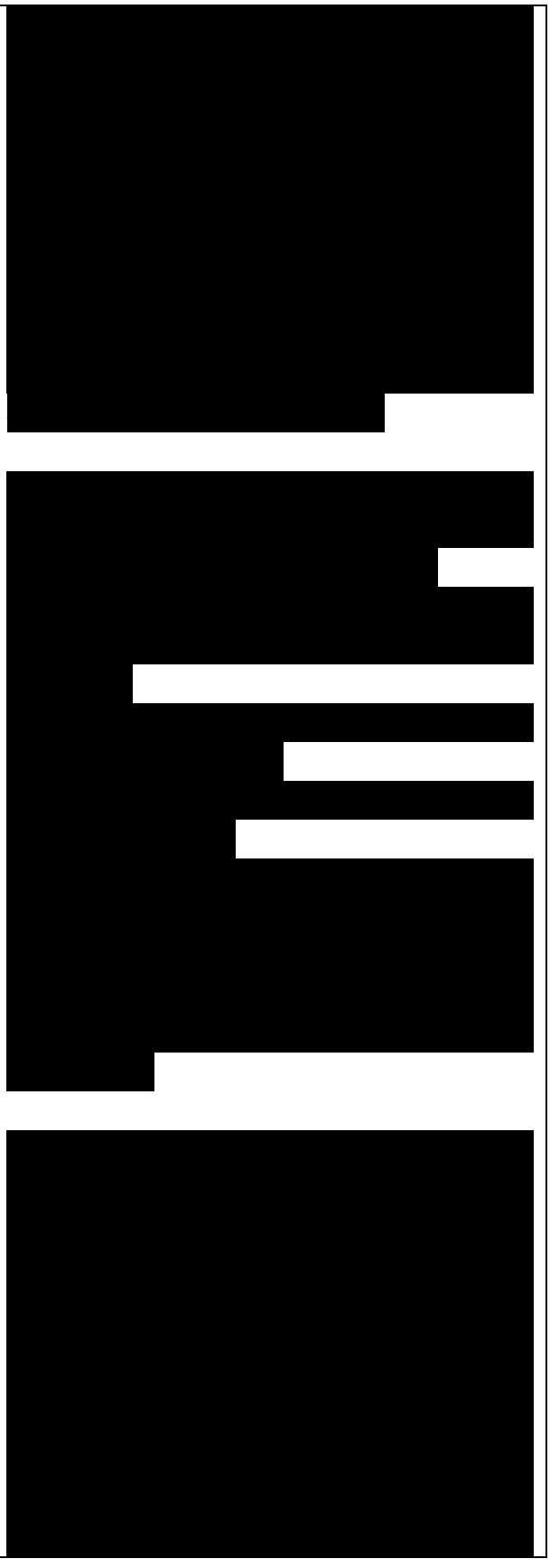
We also demonstrate how the number of negative generalized tuples changes as the minExpectedSup values change. Fig. 12 shows the results for different thresholds. One can see that the

Fig. A1. Concept hierarchies for research grant data.

Fig. A2. Concept hierarchies for credit card dataset.

number of negative generalized tuples decreases as the threshold thresholds are used to filter out the valuesets; smaller thresholds values get larger. This result is quite reasonable, because the naturally result in more valuesets.

As mentioned earlier, an interest measure is proposed to filter out uninteresting tuples. In the final experiment, we repeat 50 times to evaluate the influence of the interest measures. The experimental parameters are as follows: data size is fixed at 25000 tuples; number of attributes is 6; minValueSup is 10%; and



minExpectedSup is 2%. The run time curve with different interest measures, from 1.1 to 1.8 is illustrated in Fig. 13. Fig. 14 shows the variation of the number of negative generalized tuples. From these figures, we can see that by setting an appropriate interest measure, one can easily filter out many uninteresting negative generalized tuples and reduce the computation time.

6. Conclusion

We have proposed an effective method, the GNAOI method, for generating all negative interesting generalized knowledge at one time. The method employs the expected support concept to filter out uninteresting knowledge. In addition, three constraints and two useful closure properties are designed to reduce the search space. The method provides a simple but efficient way to carry out negative knowledge generalization from a relational database. A real-life dataset is used for experiment and the results show that the proposed method can induce more comprehensive negative generalized knowledge from relational database. There are two possible directions for future research. First, in complicated business or distributed environment, there is an urgent need for mining negative generalized knowledge from multi-databases. The proposed GNAOI method can be integrated with existing multi-database mining method [39,40] to expand its application scope. Second, it would be interesting to

extend the usage of background knowledge by using domain generalization graphs rather than concept trees or using fuzzy concept trees rather than crisp concept trees.	[redacted]